stsc

APL★PLUS® System

Nested Arrays Quick Reference Guide

This guide describes Nested Arrays features, an extension to APL. For a more detailed description of Nested Arrays operators, functions, and miscellaneous features, see Supplement Number 4 to *Enhancements in the APL★PLUS System, Version 3.0.1* (STSC Publication Number SU005). All definitions and examples below are in origin 1.

Operators

Syntax	Definition/Example	Description
`Z←L ∘.FOO R`		**Operators accept user-defined functions as arguments.**
`Z←L ∘., R`		**Operators accept all primitive functions as arguments.**
`L +.×.÷ R ↔ L (+.×).÷ R`		**Operator sequences:** operators have long left scope and short right scope.
`Z← f¨ R` `Z←L f¨ R`	`I⊃Z ↔ f I⊃R` `I⊃Z ↔ (I⊃L) f I⊃R`	**Each:** applies *f* to the items of *R*, or between corresponding items of *L* and *R*.

Nested Array Functions

Syntax	Over-strike	Definition/Example	Description
`Z←⊂R`		`0 ↔ ⍴⍴Z`	**Enclose:** turns *R* into a nested scalar; returns *R* unchanged if *R* is a simple scalar.
`Z←⊃R`		`R ↔ ⊃⊂R`	**Disclose:** converts a nested scalar into the array it contains; or returns the first item of its argument or the fill item if *R* is empty.

`Z←L⊂[K] R`		`I⊃Z ↔ (I=+\L)/[K] R`
		`⍴⍴Z ↔ 1 ; ⍴Z ↔ +/L`
		`⍴L ↔ (⍴R)[K]`
			Partitioned enclose: builds a nonsimple vector out of selected portions of an array; 1's in the Boolean vector *L* mark the beginnings of partitions of *R* to enclose.

`Z←L⊃R`		`(⍳0)⊃R ↔ R` `L⊃R ↔ (1↓L)⊃⊃R[1↑L]`	**Pick:** form of indexing for selecting items from a nested array; each item of the vector *L* specifies a layer of *R* to index through.

`Z←=R`	`=`	`0 ↔ = ⍳3`
		`1 ↔ = ⊂⍳3`
		`2 ↔ = ⊂⊂⍳3`
		`¯2 ↔ = (⊂1 2),⊂⊂3 4`
			Depth: returns number of levels of nesting, non-negative if uniform depth, negative if non-uniform.

`Z←∈R`		`⍴Z ↔ ⍴R`	**Type:** replaces characters with blanks and numbers with zeros.
`Z←L=R`	`=`		**Match:** returns 1 if the left and right arguments are identical in rank, shape, values and (if empty) datatypes (numeric or character); returns 0 otherwise.
`Z←↓[K] R`		`⍴Z ↔ (K≠⍳⍴⍴R)/⍴R`	**Split:** encloses the *K*th coordinate of *R*.
`Z←↑[K] R`		`⍴Z ↔ ((⌊K)↑⍴R),(⍴⊃R),(⌊K)↓⍴R` `R ↔ ↑[K-0.5]↓[K] R`	**Mix:** assembles items of *R* to form a new dimension between the ⌊*K*th and ⌈*K*th coordinates.
`Z←L[I]` `L[I]←Z`		`⍴Z ↔ ⍴I` `⍴⊃I ↔ ⍴⍴L`	**Indexing extended (choose):** each item of *I* has the shape of the rank of *L* and forms a valid index to *L* (same form as argument to pick); items of *L* are indexed in or out.

Existing Functions and Operators on Nested Arrays

Syntax	Definition/Example	Description
`Z← f R` `Z←L f R`	`I⊃Z ↔ f I⊃R` `I⊃Z ↔ (I⊃L) f I⊃R`	**Scalar functions:** pervasive through nested arrays; they apply through every level of nesting. In the dyadic form, corresponding levels of nesting must be rank, shape, and type conformable.
`Z← f/ R`	`Z←⊂(⊃R) f ⊃f/ 1↓R` † or `Z←⊂(1⊃R)f (2⊃R)f ...`	**Reduction:** operates between the items of the array to produce items of the result. For a scalar function f on simple arrays, there is no change in behavior.
`Z← f\ R`	`Z[I] ↔ f/I↑R` †	**Scan:** the redefinition of reduction affects the behavior of scan. For a scalar function f on simple arrays, there is no change in behavior.
`Z←L f.g R`	`Z← f/ L g¨ R` †	**Inner product:** both f and g may be nonscalar functions. For scalar functions f and g on simple arrays, there is no change in behavior.
`Z←L ∘.g R`	`(I,¨J)⊃Z ↔ ⊂(I⊃L)g J⊃R`	**Outer product:** g operates between the items of L and R. For a scalar function g on simple arrays, there is no change in behavior.
`Z←L,[K] R`		**Catenation/lamination:** either or both arguments may be nonsimple. Arguments may differ in type.
`L⍳R  L∈R`		**Dyadic iota and epsilon:** substitute ⍨ for = in their definitions.

`L⍴R  ⍴R  L⌽R`
`⌽R  L/R  L\R`
`L⌽R  ⌽R  L↑R`
`L↓R  ,R  →R`
		Miscellaneous functions: the right argument may be nonsimple; the left argument must be simple. Fill items for $L↑R$ and $L\R$ are given by $∈⊃R$.

`L⊥R  L⊤R  L⊞R`
`⊞R  ⍋R  ⍒R`
`L⍒R  ⍒R  ⍎R`
`L?R`
		Functions not defined on nested arrays: these functions are defined on simple arrays only.

† Definition given for vectors; function extends to higher-dimensional arrays.

Miscellaneous Features

Heterogeneous arrays: arrays that can contain both characters and numbers. Only simple (non-nested) arrays can be heterogeneous. For example, `'ABC',⍳4`.

Strand notation: a means of entering vectors, either simple or nested. Any position in a vector can now consist of a number or character (as before), any rank or shape array, or an expression. For example, `A (2 3) ('BAT')`.

Strand notation assignment: allows more than one variable to be assigned in one operation. For example, `A B C←1 2 3` same as `A←1 ⋄ B←2 ⋄ C←3`. `A B←B A` exchanges `A` and `B`.

STSC, INC.
2115 East Jefferson Street
Rockville, Maryland 20852
301-984-5000

Printed in the U.S.A.

APL*PLUS is a service mark and trademark of STSC, Inc., registered in the United States Patent and Trademark Office.

Copyright © 1982 STSC, Inc.

H028-0882